

3D Semantic Representation of Actions from efficient stereo-image-sequence segmentation on GPUs

Alexey Abramov Eren Erdal Aksoy Johannes Dörr
Florentin Wörgötter
Georg-August University, BCCN Göttingen, III Physikalisches Institut
Göttingen, Germany
`{abramov,eaksoye,jdoerr,worgott}@bccn-goettingen.de`

Karl Pauwels
Laboratorium voor Neuro- en Psychofysiologie
K.U.Leuven, Belgium
`karl.pauwels@med.kuleuven.de`

Babette Dellen
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Barcelona, Spain
`bdellen@iri.upc.edu`

Abstract

A novel real-time framework for model-free stereo-video segmentation and stereo-segment tracking is presented, combining real-time optical flow and stereo with image segmentation running separately on two GPUs. The stereo-segment tracking algorithm achieves a frame rate of 23 Hz for regular videos with a frame size of 256×320 pixels and nearly real time for stereo videos. The computed stereo segments are used to construct 3D segment graphs, from which main graphs, representing a relevant change in the scene, are extracted, which allow us to represent a movie of e.g. 396 original frames by only 12 graphs, each containing only a small number of nodes, providing a condensed description of the scene while preserving data-intrinsic semantics. Using this method, human activities, e.g., handling of objects, can be encoded in an efficient way. The method has potential applications for manipulation action recognition and learning, and provides a vision-front end for applications in cognitive robotics.

1. Introduction

Movies contain abundant information about the visual scene, which, if choosing the raw signals for representations, render the application of any logic or learning scheme intractable. This problem occurs for example if we want

to understand and learn both courses and consequences of manipulations from visual data. A reduced representation of the scene is urgently required to make such problems tractable by algorithms operating on a small number of abstract descriptors (symbols). Finding this reduced representation without prior knowledge on the data (model free) thus represents a major challenge in cognitive-vision applications – this problem is also known as the signal-symbol gap [9].

In this paper, we present a novel framework for bridging this gap. We aim at creating a condensed description of stereo movies by computing the 3D relations between tracked image segments for generating 3D semantic graphs, which, in the future, will allow us to encode manipulation actions in an efficient way. To achieve this goal, three main problems need to be solved: (i) Stereo images need to be segmented in real time in a consistent way and image segments need to be tracked along the movie, i.e., segments representing the same part of an object should carry the same label all the time. Note that image segmentation represents a logical step towards our goal because redundant information is thereby grouped and condensed into higher level representational entities (segments). The method should be entirely data driven. (ii) The 3D relations of the segments need to be derived with sufficient accuracy, and graphs need to be constructed. Then, only the graphs representing a relevant change in the scene should be ex-

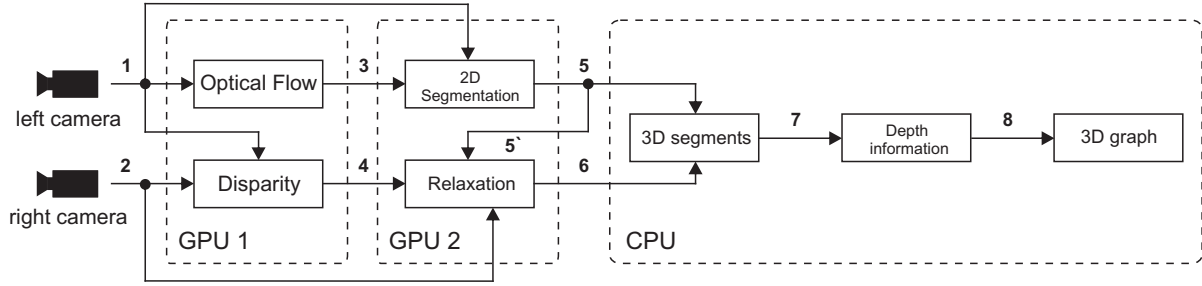


Figure 1. The architecture of 3D segment tracking framework consists of a stereo camera, two GPUs, one CPU, and various processing components that are connected by channels in the framework. Output data of all components can be accessed from any component in the framework.

tracted, further removing redundant information. (iii) The algorithms should run in real time or close to real-time¹ to allow the framework to be used for robotic applications.

Several approaches for video segmentation have been proposed in the past, where some methods rely on segmenting each frame independently, followed by a segment matching step based on their low-level features [15, 7], while other methods use motion projection to link segments [12, 19]. It was shown recently that image segments can be tracked along the frames of a movie in a model-free way, i.e. without assuming a data model of some kind, using the method of superparamagnetic clustering of data [5].

Real-time requirements render the video segmentation algorithms currently inadequate for most robotic applications. Furthermore, stereo movies have not been treated by any of these works. To overcome these limitations, we developed real-time model-free image segmentation on GPUs based on a novel parallel method, combined with real-time phase-based optical flow [13] and stereo [14], executed on GPU as well, for segment tracking.

The framework will potentially be applicable to a wide range of problems in the field of action and manipulation recognition and learning, and may serve as vision-front end in cognitive robots.

2. 3D segment tracking framework

2.1. Overview

The architecture of the 3D segment tracking framework is shown in Fig. 1. The left and right images from a stereo camera enter into the framework via channels 1 and 2 respectively. Optical flow and disparity are computed on GPU 1 using a real-time algorithm [13], and the results are accessible from channels 3 and 4 respectively (see Section 2.3).

¹By real-time we understand processing of a full frame at 25Hz or faster.

For the images from the left camera, the labels from a previous segmentation are warped to the current frame using optical flow (channel 3). The new label configuration is used as an initialization for the real-time segmentation algorithm running on GPU 2 (see Section 2.2-2.4). This way, the required time for label relaxation can be reduced, and, even more importantly, a consistent labeling of the frames can be achieved, i.e. segments describing the same object part are likely to carry the same label (segment tracking) (see Section 2.4). The results of the segmentation can be accessed from channel 5 and used to compute the label initialization for the segmentation of the right frame via channel 5' (see Section 2.5). This time, the labels are warped using phase-based disparity information obtained from channel 4. The segmentation result of the right image, which is now consistently labeled with respect to the images from the left camera, is stored in channel 6. Segments larger than a predefined threshold are extracted and stored in channel 7. The stereo-segment correspondences are used to find the 3D structure of the segments using a recent stereo method on the CPU [6] and stored in channel 8 (see Section 2.6). In a final step, relevant information about the 3D structure of the segments is extracted from the computed disparity map and used together with the segmentation results to construct an abstract 3D description of the scene, which is represented by undirected graphs in which nodes and edges represent 3D segments and their neighborhood relations (see Section 2.7) as proposed by Aksoy et al. (2010) [2].

2.2. Image segmentation algorithm

The method of superparamagnetic clustering solves the segmentation problem by finding the equilibrium states of the energy function of a ferromagnetic Potts model in the superparamagnetic phase [4, 11, 18]. The Potts model describes a system of interacting granular ferromagnets or spins that can be in q different states, characterizing the pointing direction of the respective spin vectors. Three

phases, depending on the system temperature, i.e. disorder introduced to the system, are observed: the paramagnetic, the superparamagnetic, and the ferromagnetic phase. In the ferromagnetic phase, all spins are aligned, while in the paramagnetic phase the system is in a state of complete disorder. In the superparamagnetic phase regions of aligned spins coexist. Blatt et al. (1998) applied the Potts model to the image segmentation problems in a way that in the superparamagnetic phase regions of aligned spins correspond to a natural partition of the image data [4]. Finding the image partition corresponds to the computation of the equilibrium states of the Potts model.

The equilibrium states of the Potts model have been approximated in the past using the Metropolis-Hastings algorithm with annealing [8] and methods based on cluster updating, which are known to accelerate the equilibration of the system by shortening the correlation times between distant spins, such as Swendsen-Wang [17], Wolff [20], and energy-based cluster updating (ECU) [11, 18]. All of these methods obey detailed balance, ensuring convergence of the system to the equilibrium state. Here we achieve efficient performance using the Metropolis algorithm with annealing [8], which can be easily parallelized and implemented on a GPU. The method further has the advantage that results from a previous segmentation can be utilized by the algorithm to find a consistent segmentation of the next frame within a small number of Metropolis updates only, drastically reducing computation time.

The real-time image segmentation algorithm proceeds as follows. In the Potts model, a spin variable σ_k , which can take on q discrete values v_1, v_2, \dots, v_q , called spin states, is assigned to each pixel of the image. The energy of the system is described by

$$E = - \sum_{\langle ij \rangle} J_{ij} \delta_{ij} \quad , \quad (1)$$

with the Kronecker sign

$$\delta_{ij} = \begin{cases} 1 & \text{if } \sigma_i = \sigma_j, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

where σ_i and σ_j are the respective spin variables of two neighboring pixels i and j . The function

$$J_{ij} = 1 - |\mathbf{g}_i - \mathbf{g}_j| / \bar{\Delta} \quad (3)$$

is a coupling constant, determining the interaction strength, where \mathbf{g}_i and \mathbf{g}_j are the respective color vectors of the pixels, and

$$\bar{\Delta} = \alpha \cdot \left(\sum_{\langle i,j \rangle} |\mathbf{g}_i - \mathbf{g}_j| / \sum_{\langle i,j \rangle} 1 \right) \quad (4)$$

computes the averaged color vector difference of all neighbors $\langle i, j \rangle$. The factor $\alpha \in [0, 10]$ is a system parameter.

The Metropolis algorithm allows generating spin configurations S which obey the Boltzmann probability distribution

$$P(S) \sim \exp[-\beta E(S)] \quad , \quad (5)$$

where $\beta = 1/kT$, T is the temperature parameter, and k is the Boltzmann constant.

Initially, values are assigned randomly to all spin variables. According to the Metropolis algorithm, each spin-update procedure consists of the following steps [10]:

1. The system energy E_A of the current spin configuration S_A is computed according to Eq. 1.
2. A pixel i with spin variable σ_i in spin state v_l is selected and for each possible move to a new spin state $\sigma_i \neq v_l$ the energy E_B of the resulting new spin configuration S_B is computed according to Eq. 1. The number of possible moves is $(q - 1)$.
3. Among all new possible configurations we find the configuration with the minimum energy

$$E_{new} = \min(E_1, E_2, \dots, E_{q-1}) \quad , \quad (6)$$

and compute the respective change in energy

$$\Delta E = E_{new} - E_A \quad . \quad (7)$$

4. If the total energy of the configuration is decreased by this move, i.e. $\Delta E < 0$, the move is always accepted.
5. If the energy increased, i.e. $\Delta E > 0$, the probability that the proposed move will be accepted is given by

$$P_{A \rightarrow B} = \exp\left(-\frac{|\Delta E|}{kT_n}\right) \quad , \quad (8)$$

and

$$T_{n+1} = \gamma T_n \quad \gamma < 1 \quad , \quad (9)$$

where γ is the annealing coefficient. We draw a number ξ randomly from a uniform distribution in the range of $[0, 1]$. If $\xi < P_{A \rightarrow B}$, the move is accepted.

Each spin update involves only the nearest neighbors of the considered pixel. Hence, spin variables of pixels that are not neighbors of each other can be updated simultaneously [3]. Therefore the Metropolis algorithm fits very well to the GPU architecture. For the first frame of the image sequence, a short-cut via a pre-segmentation step, is used to allow sufficiently fast segmentation. More details can be found in [1]. In the experiments we will use a large number of spins, which then serve as a segment label. Note that these wordings are thus equivalent here.

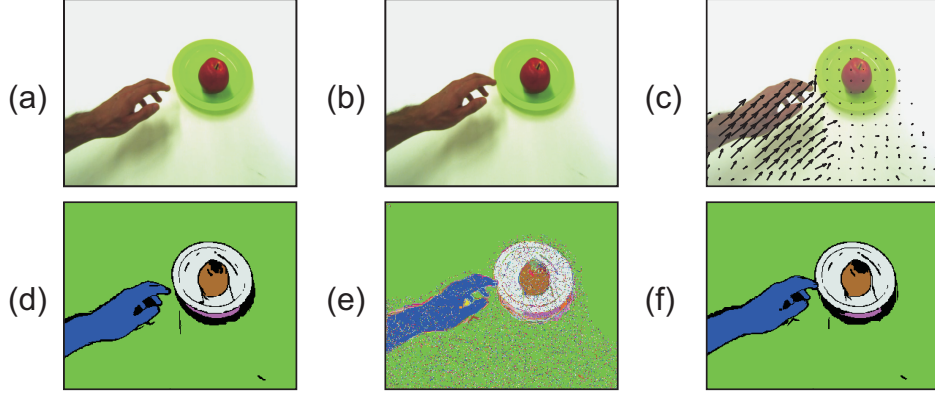


Figure 2. Segmentation of two adjacent frames in a sequence. (a) Original frame $\{t\}$. (b) Original frame $\{t+1\}$. (c) Estimated optical flow field from phase-based method (subsampled 16 times and scaled 5 times). (d) Extracted segments S_t for frame $\{t\}$. (e) Initialization of frame $\{t+1\}$ after the spin transfer. (f) Extracted segments S_{t+1} for frame $\{t+1\}$.

2.3. Phase-based optical flow and disparity

Since fast processing is a very important issue in our work, we use the GPU-based real-time optical flow algorithm proposed by Pauwels et al. (2008) [13]. This algorithm belongs to the class of phase-based techniques, which are highly robust to changes in contrast, orientation and speed. This particular algorithm integrates the temporal phase gradient (extracted from five subsequent frames) across orientation and gradually refines its estimates by traversing a Gabor pyramid from coarser to finer levels. In our framework optical flow is computed for the left video stream only. The algorithm provides a vector, at each pixel indicating its motion

$$\mathbf{u}(\mathbf{x}, \mathbf{y}) = (u_x(x, y), u_y(x, y)) \quad , \quad (10)$$

This allows us to link pixels of two subsequent frames $\{t\}$ and $\{t+1\}$ (see Fig. 2). In order to link pixels of correspondent left and right frames displacements for correspondent left and right pixels have to be estimated. For this purpose the disparity algorithm can be used. Our system relies on rectified images and therefore only horizontal disparities need to be determined. The rectification is done in real-time with a fixed stereo-geometry. The disparity algorithm that is used in our framework is also phase-based and operates on phase differences between the left and right image as opposed to temporal phase gradients in optical flow algorithm. It is described in more detail in [14].

2.4. Image-sequence segmentation using optical flow based label warping

We obtain information about pixel correspondences between subsequent frames via a phase-based optical flow algorithm applied to the frame sequence, as described in 2.3.

Since we are using a local algorithm, optical flow cannot be estimated everywhere, for example not in weakly-textured regions. For pixels in these regions, vertical and horizontal flows, i.e. u_y and u_x , do not exist. Thus we make an assumption that these pixels did not change position between frames $\{t\}$ and $\{t+1\}$, i.e. $u_y = 0$ and $u_x = 0$. We find the new label configuration by translating all labels according to the derived flow maps.

Suppose frame $\{t\}$ is segmented and S_t is its final label configuration (see Fig. 2(d)). The labels can be transferred from frame $\{t\}$ to frame $\{t+1\}$ according to

$$S_{t+1}(x_{t+1}, y_{t+1}) = S_t(x'_t, y'_t) \quad , \quad (11)$$

where

$$x'_t = x_{t+1} - u_x(x_t, y_t) \quad (12)$$

$$y'_t = y_{t+1} - u_y(x_t, y_t) \quad . \quad (13)$$

Labels which did not obtain an initialization via Eq. 11 are then given a randomly chosen label between 1 and q . Once frame $\{t+1\}$ is initialized (see Fig. 2(e)), a relaxation process is needed in order to fix erroneous bonds that can take place during the spin states transfer (see 2.2). We found that 20 additional Metropolis updating iterations are sufficient to obtain satisfactory segmentation results (see Fig. 2(d,f) where correspondent segments are labeled by identical colors).

2.5. Stereo segmentation using disparity-based label warping

Segmentation of every right frame is obtained in a similar way. Here, label initialization is obtained by translating the labels from the segmentation of the left image using the

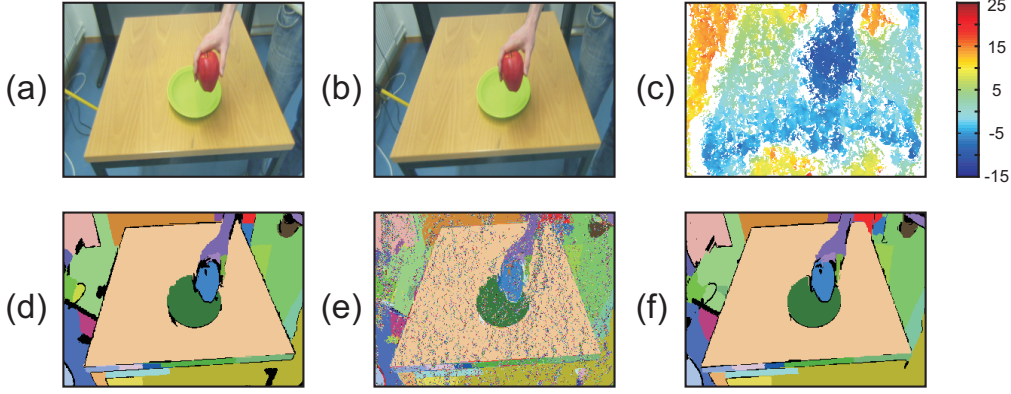


Figure 3. Segmentation of a stereo pair. (a) Original left frame $\{t_L\}$. (b) Original right frame $\{t_R\}$. (c) Estimated disparity map from phase-based stereo. (d) Extracted segments S_L^t for frame $\{t_L\}$. (e) Initialization of frame $\{t_R\}$ after the spin transfer. (f) Extracted segments S_R^t for frame $\{t_R\}$.

sparse disparity map D derived from phase-based stereo. Since the stereo algorithm relies on phase (and not magnitude), it is sufficient for successful matchings even when objects are weakly textured (see Fig. 3(a,b,c)). However it cannot provide information for all pixels. For this reason, we make the assumption that pixels from frame $\{t_L\}$ having no correspondence in frame $\{t_R\}$ have zero displacement, i.e. $D(x^t) = 0$.

We suppose that the left frame $\{t_L\}$ is segmented and S_L^t is its final label configuration (see Fig. 3(d)). Labels are transferred from frame $\{t_L\}$ to frame $\{t_R\}$ according to

$$S_R^t(x_R^t, y_R^t) = S_L^t(x_L^t, y_L^t) \quad , \quad (14)$$

where

$$x_L^t = x_R^t - D(x_L^t, y_L^t) \quad (15)$$

$$y_L^t = y_R^t \quad . \quad (16)$$

Pixels that did not obtain a label initialization this way are given a randomly chosen label between 1 and q (see Fig. 3(e)). Once frame $\{t_R\}$ is initialized, again a relaxation process is needed in order to fix erroneous bonds (see 2.2). We found that about 50 – 100 additional Metropolis updating iterations are sufficient to obtain satisfactory 3D segmentation results (see Fig. 3(d,f)) where correspondent segments are labeled by identical colors).

2.6. Disparity from stereo-segment correspondences

In weakly-textured scenes, segment correspondences can be used to derive the disparity map of the scene in situations where texture-based methods are bound to fail [6]. Following the method proposed by [6], stereo segments are com-

puted first using our framework, then, reliable disparity estimates from segment boundaries and weak inner-segment texture are extracted together with an occlusion map, which is derived from the approximate depth ordering of the stereo segments. Finally the information is fused and a segment-constrained interpolation algorithm is employed to obtain a dense disparity map (see Fig. 4(c) and Fig. 5(c)). The disparity map can then be used to establish 3D relations between segments in the graphical representation (see 2.7). We further defined a confidence value for each segment by summing the input confidence values (before interpolation - see [6]) for the respective segment and dividing it by the total size of the segment. Large segments for which little inner-segment disparities or edge disparities could be found will receive a lower confidence than smaller segments for which, e.g., the disparities are known along the whole segment boundary. Using this approach, we found that for the large white table of the sequences shown in Fig. 4(a) and Fig. 5(a), disparities could be interpolated only with very low confidence. This is because most of the boundaries of the table are cut off by the frame boundaries and because the table has no texture. As a consequence, stereo cannot provide sufficient depth information here.

2.7. 3D Semantic Scene Graphs

Once 3D segments are extracted, we represent the scene by undirected and unlabeled semantic graphs [2]. The graph nodes are the segment labels and plotted at the center of each segment. The nodes are then connected by an edge if the segments are neighbors and their depth differences are less than a predefined threshold value (see Fig. 4(d) and Fig. 5(d)). We also define a 3D field of view boundary and ignore segments whose depth value does not exceed this

boundary. With this method we create a focus of attention and ignore objects outside the 3D boundary, such as human like perception.

In the temporal domain, 3D scene graphs represent spatial relations between nodes. Unless spatial relations change, the scene graphs remain topologically the same. The only changes in the graph structures are the node positions or the edge lengths depending on the object trajectory and speed. Consequently, any change in the spatial relation between nodes corresponds to a change in the main structure of the scene graphs. Therefore, those changes in the graphs can be employed to define action primitives. Considering this fact, we apply an exact graph-matching method in order to extract the main graphs by computing the eigenvalues and eigenvectors of the adjacency matrices of the 3D graphs [16]. A change in the eigenvalues or eigenvectors then corresponds to a structural change of the graph. As experimental data we use two real stereo-image sequences (see Section 3). The total frame number of the first image sequence “Making a sandwich” is 396, however, after extracting the main graphs, only 12 frames are left, each defining a single action primitive. Due to page limitations Fig. 4(d) shows only 7 of the main graphs. In the second image sequence we have interleaved chained manipulations, i.e., “Cutting a salami”, “Making a sandwich”, and “Putting on a plate”, making a total of 2977 frames. In Fig. 5(d), 6 sample main graphs are shown, each representing an action primitive.

2.8. Experimental environment

As hardware platforms for our 3D segment tracking framework we use one NVIDIA card GeForce GTX 295 (with 896 MB device memory) consisting of two GPUs each of which has 30 multiprocessors and 240 processor cores in total and CPU 2.2GHz AMD Phenom Quad 9550 (using a single core) with 2 GB RAM. We use the first GPU of the card to calculate optical flow and disparity. The second GPU is mainly used for frame-wise image segmentation. Using two GPUs allows us to run some parts of the framework physically in parallel and achieve better processing time as compared to having one GPU only.

3. Experimental Results

The developed framework is applied to two real stereo-image sequences: “Making a sandwich” (see Fig. 4(a)) and a sequence consisting of interleaved chained actions, i.e., “Cutting a salami”, “Making a sandwich”, and “Putting on a plate” (see Fig. 5(a)). In the first example (see Fig. 4(a)) two arms are appearing in the scene, putting the salami and cheese slices on a piece of bread, and then leaving the scene. The second sequence (see Fig. 5(a)) consists of two arms that are first taking a bread from a toaster, putting a piece

of a cheese on it, and then cutting off a slice of salami with a knife. After putting the salami on top of the cheese, the sandwich is being placed on a plate and the arms are leaving the scene. Respective image segments of some sample frames from the left sequences are given in Fig. 4(b) and 5(b). Dense disparity maps obtained for extracted stereo segments are given in Fig. 4(c) and 5(c). The low-confidence-value area of the table segment is depicted with a black color in the dense disparity maps (see Section 2.6). Fig. 4(d) and 5(d) illustrate 3D semantic scene graphs of the selected frames. The graphs show that all relevant object parts in the scene can be represented by unique labels and tracked during the whole image sequence. Moreover, each graph defines a topological change in the scene. For instance, in frame number 112 of the first image sequence (see Fig. 4(d)) we observe that the arm represented by graph node number 113 has an edge only with the table (graph node number 3). In 2D, the arm would have an edge with the cheese since their respective segments are 2D neighbors, but this edge is ignored because the depth difference between the arm and the cheese is too large. In the next main graph (frame number 167) an edge connects the arm (node number 113) with the salami (node number 247) because the segments are touching in 3D.

For mono-image sequences with a frame size of 256×320 pixels a processing time of 23 Hz was achieved that demonstrates the applicability of the framework to mono-video processing tasks in real-time or close to real-time. However, for sequences containing weakly-textured regions more additional Metropolis updating iterations are needed in order to achieve a final stable configuration (see 2.4). For the stereo video segmentation with the same frame size the maximum processing time that can be achieved for the moment is 10 Hz. However this is not the case for all stereo-image sequences.

4. Discussion

In this work we presented a novel highly parallel framework for deriving a condensed 3D semantic representation of visual scenes based on a near real-time segment tracking procedure implemented in parallel on GPUs. We achieved (i) stereo image-sequence segmentation and segment tracking in near real time, and, in a subsequent step, (ii) to build 3D semantic graphs using a recent stereo method for defining the 3D segment relations and exact graph matching for the extraction main graphs (action primitives). Algorithmic parts for (ii) are implemented on CPU and are not yet running in real-time, but potential parallel solutions are currently investigated. The main contribution of this work is the provision of a novel framework for representing image sequences in an efficient way, which may serve as vision-front end for cognitive robots in the future. Most importantly, the segment tracking is entirely data driven (model-

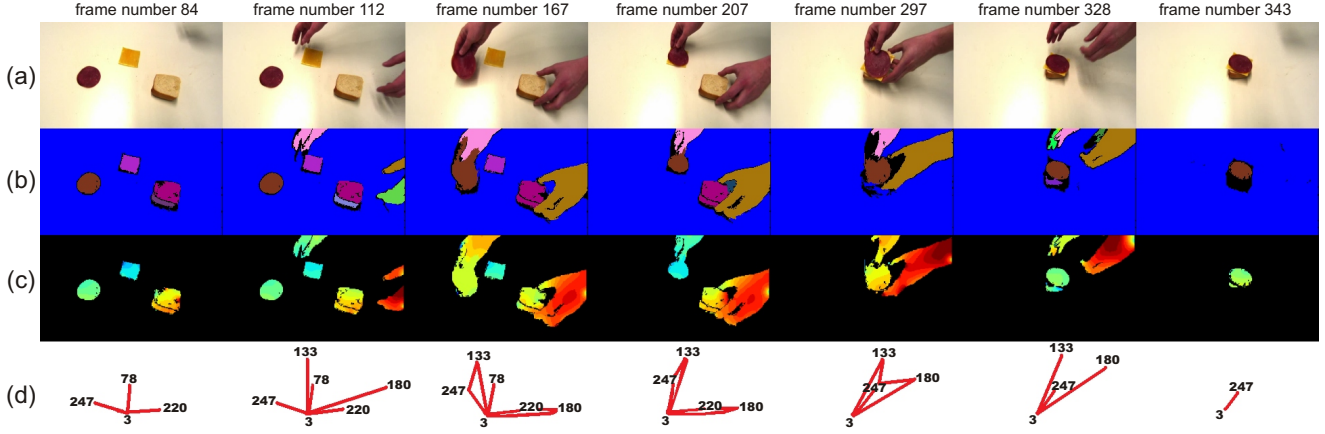


Figure 4. Results for the sample action “Making a sandwich”. (a) Original frames from the left image sequence. (b) Extracted segments for frames of the left sequence. (c) The dense disparity maps obtained for extracted stereo segments. The disparity values are color-coded from blue (small) to red (large). Areas of low confidence are colored black, i.e., the uniform and untextured area of the table, for which only poor disparity results could be obtained. (d) Final 3D semantic scene graphs, representing action primitives.

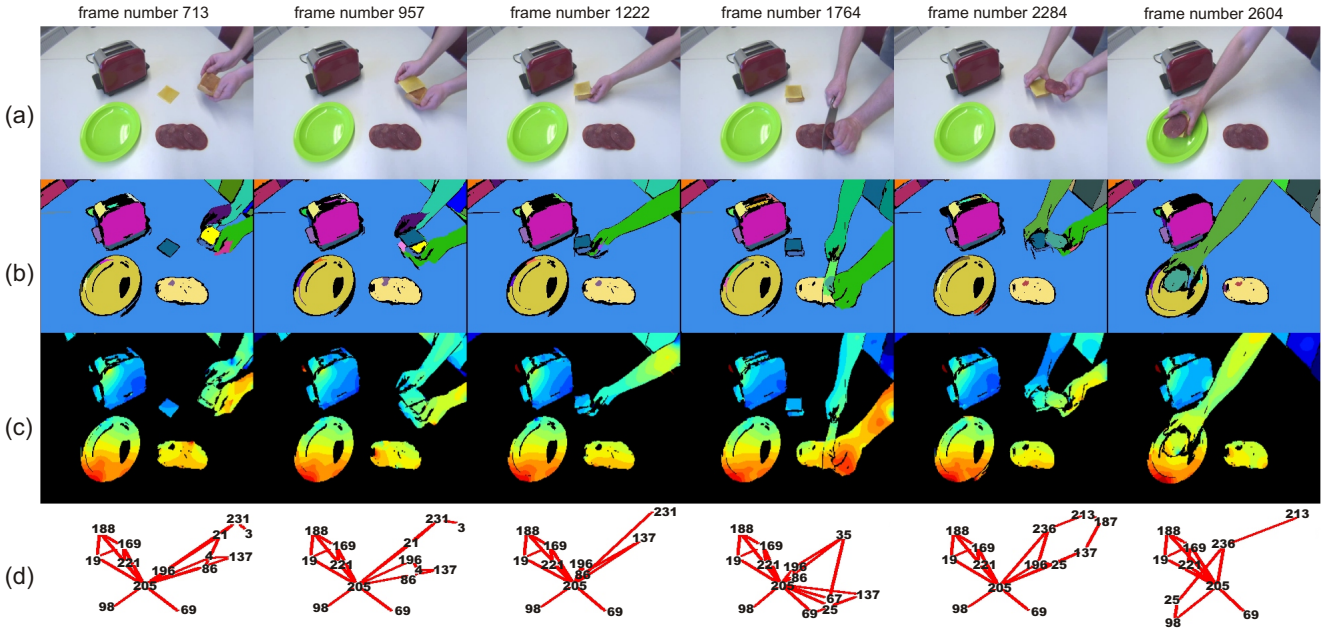


Figure 5. Results for the interleaved sample actions, i.e. “Cutting a salami”, “Making a sandwich”, and “Putting on a plate”. (a) Original frames from the left image sequence. (b) Extracted segments for frames of the left sequence. (c) The dense disparity maps obtained for extracted stereo segments. The disparity values are color-coded from blue (small) to red (large). Areas of low confidence are colored black, i.e., the uniform and untextured area of the table, for which only poor disparity results could be obtained. (d) Final 3D semantic scene graphs, representing action primitives.

free) and thus does not require prior data models to be provided, making the method more robust and more widely applicable. To our knowledge, the quite extensive framework presented in this paper is the first of its kind. Although video segmentation has been proposed before, the methods

used therein are often model based, or not running real time [15, 7, 12, 19, 5]. Furthermore, our segment tracking procedure is developed for a specific goal, i.e. the construction of 3D semantic graphs, which has not been attempted in this way before.

The proposed framework has been applied to several real stereo-image sequences. Obtained results demonstrate stability of the segment tracking procedure both for mono-image sequences and for stereo-image sequences. For the segmentation of mono-image sequences with a frame size of 256×320 pixels we obtained processing time sufficient for real-time or close to real-time applications. Stereo-image sequence segmentation is not real time yet, but the developed framework is sufficiently fast for being applicable to robot-real-time 3D applications.

The algorithm has some weak points. At the moment, relaxation times increase if two previously disjoint regions ought to be joined. The processing time of stereo-video segmentation also depends on texture information and the size of occluded areas. For sequences with weakly textured areas and relatively large occlusions a longer relaxation process is required to label those areas. As the method used for the pre-segmentation takes into account only color information of interacting pixels, the developed algorithm has a better performance for large segments than for small ones, since the color segmentation works best for large uniform image regions. For textured areas, corresponding to small regions, the performance of our algorithm decreases, because the gray-value similarity of neighboring pixels is too low. Towards better results for very textured or noisy image sequences, in the future texture segmentation can be incorporated into the algorithm.

In the future, we aim to apply the developed framework to tasks in cognitive robotics. Since each main graph represents an action primitive and also the respective object relations, we are planning to use this information to recognize and classify the manipulation actions and categorize objects based on their role during the manipulation.

5. Acknowledgements

The work has received support from the BMBF funded BFNT Göttingen, Grant Project 3a. B.D. also acknowledges support from Spanish Ministry for Science and Innovation via a Ramon y Cajal fellowship.

References

- [1] A. Abramov, T. Kulvicius, F. Wörgötter, and B. Dellen. Real-time image segmentation on a gpu. In *Facing the Multicore-Challenge*, Heidelberg, Germany, 2010. [3](#)
- [2] E. E. Aksoy, A. Abramov, F. Wörgötter, and B. Dellen. Categorizing object-action relations from semantic scene graphs. In *IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, 2010. [2, 5](#)
- [3] G. T. Barkema and T. MacFarland. Parallel simulation of the Ising model. *Physical Review E*, 50(2):1623–1628, 1994. [3](#)
- [4] M. Blatt, S. Wiseman, and E. Domany. Superparamagnetic clustering of data. *Physical Review Letters*, 76(18):3251–3254, 1996. [2, 3](#)
- [5] B. Dellen, E. E. Aksoy, and F. Wörgötter. Segment tracking via a spatiotemporal linking process including feedback stabilization in an n-d lattice model. *Sensors*, 9(11):9355–9379, 2009. [2, 7](#)
- [6] B. Dellen and F. Wörgötter. Disparity from stereo-segment silhouettes of weakly-textured images. In *British Machine Vision Conference*, London, UK, 2009. [2, 5](#)
- [7] Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:800–810, 2001. [2, 7](#)
- [8] D. Geman and S. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):721–741, 1984. [3](#)
- [9] P. König and N. Krüger. Perspectives: Symbols as self-emergent entities in an optimization process of feature extraction and predictions. *Biological Cybernetics*, 94(4):325–334, 2006. [1](#)
- [10] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. of Chem. Phys.*, 21(11):1087–1091, 1953. [3](#)
- [11] R. Opara and F. Wörgötter. A fast and robust cluster update algorithm for image segmentation in split-lattice models without annealing - visual latencies revisited. *Neural Computation*, 10(6):1547–1566, 1998. [2, 3](#)
- [12] L. Patras, E. A. Hendriks, and R. L. Legendijk. Video segmentation by map labeling of watershed segments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:326–332, 2001. [2, 7](#)
- [13] K. Pauwels and M. Van Hulle. Realtime phase-based optical flow on the gpu. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Computer Vision on the GPU*, Anchorage, Alaska, 2008. [2, 4](#)
- [14] S. Sabatini, G. Gastaldi, F. Solari, J. Diaz, E. Ros, K. Pauwels, M. Van Hulle, N. Pugeault, and N. Krüger. Compact and accurate early vision processing in the harmonic space. In *International Conference on Computer Vision Theory and Applications*, pages 213–220, Barcelona, 2007. [2, 4](#)
- [15] P. Salembier and F. Marques. Region-based representations of image and video: Segmentation tools for multimedia services. *IEEE Trans. Circuits Syst. Video Technol.*, 9:1147–1169, 1999. [2, 7](#)
- [16] M. F. Sumsi. *Theory and Algorithms on the Median Graph. Application to Graph-based Classification and Clustering*. PhD thesis, Universitat Autònoma de Barcelona, 2008. [6](#)
- [17] R. Swendsen and S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 76(18):86–88, 1987. [3](#)
- [18] C. von Ferber and F. Wörgötter. Cluster update algorithm and recognition. *Physical Review E*, 62(2):1461–1464, 2000. [2, 3](#)
- [19] D. Wang. Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 8:539–546, 1998. [2, 7](#)
- [20] U. Wolff. Collective Monte Carlo updating for spin systems. *Physical Review Letters*, 62(4):361–364, 1989. [3](#)